Supplemental Methods of Integrating SNVs and CNAs on a phylogenetic tree from single-cell DNA sequencing data

Liting Zhang[1], Hank W. Bass[1], Jerome Irianto[1], and Xian Mallory[1]

Florida State University, Tallahassee FL 32306, USA
xfan2@fsu.edu

1 Description of our simulator

To fully examine the performance of SCsnvcna, we simulated 15 sets of data, each with a varying variable shown in Supplemental Table S1.

We used a simplified and modified version of SimSCSnTree(Mallory and Nakhleh, 2022) to simulated these datasets. Unlike SimSCSnTree, the simulator we designed in this study does not simulate the actual CNA and SNV's locations. Neither does it simulate the sequencing reads. Instead, it simulates the cell by SNV matrix in which the cells and SNVs are represented by the rows and columns, respectively. In addition, we simulated an inferred CNA tree structure and CNA cell cellular prevalence (CP) on the inferred CNA tree. The inferred CNA tree structure along with its CNA cell CPs and the cell by SNV matrix serve as the inputs to SCsnvcna. In addition, we modified SimSCSnTree such that the simulator can simulate the loss of SNVs due to copy number loss, and such loss of SNVs will be reflected in the cell by SNV matrix.

The following describes our simulator in detail. The whole process is to mimic the biological process of the CNAs and SNVs growing on a tree, as well as the sequencing process of obtaining SNVs from SNV cells and CNAs from CNA cells along with an inferred CNA tree. First, we simulate a clonal tree on which each leaf node represents a subclone of cells. This clonal tree serve as the ground truth phylogenetic tree (the so called "**true tree**"). The number of subclones is predetermined and can be varied (Experiment 4 in **Supplemental Table S1**). The tree structure is simulated by a Beta-splitting model. In the Beta splitting model, every node on the tree has an interval, based on which both SNV cells' CPs and CNA cells' CPs will be simulated. Thus we call the length of the intervals the "**true CPs**". The interval has to be within [0, 1]. Root's interval is [0, 1]. Such an interval represents the size of the clone for each node. The tree structure is generated in the following way. Starting from a normal node which is the root and temporarily a leaf node, for every round, our simulator selects a leaf node to split to two daughter nodes until the number of leaf nodes reaches the desired number. For the first round, the simulator has no other choice but to select the root to split. In the following rounds, the chance that a leaf node is chosen to split equals the length of its interval. When a node splits, it is no longer a leaf node but becomes an internal node. Its two daughter nodes will become the leaf nodes. The two daughter nodes will split the interval that the parent node has according to a Beta distribution, whose alpha value is fixed to be 0.5, and beta value is one of the varying variables that we use in the examination of SCsnvcna (called "Beta splitting variable", Experiment 9 in **Supplemental Table S1**). Beta splitting variable controls the contrast of the subclone sizes on the leaf level. The larger the Beta splitting variable, the more similar the subclone CPs are to each other and thus the more challenging for SCsnvcna to place SNV cells and SNVs correctly. Branch lengths of the tree follow exponential distributions whose λ value is set to be 1.

Given the simulated true tree structure, we then simulate CNAs on the true tree. Since the focus of SCsnvcna is to place a SNV on a CNA tree, our focus is not on CNAs except where a CNA may cause SNV loss. Thus we do not simulate the actual CNAs. Instead, for all experiments except experiment 13 (**Supplemental Table S1**, **Supplemental Methods Section 2.4**), for each edge on the true tree, we add a CNA. Such a CNA has only an index, but not the actual information such as chromosome, start, end, copy number loss or gain, and how many copies are lost/gained. The reason that we add only one CNA on each edge is because one CNA, if can be detected in the real case, is enough to distinguish the left and right child subclones. In the real case scenario, it is possible that not every edge on the true tree has at least one CNA. This means that the true tree may branch by only SNVs but not CNAs, leading to two subclones of cells distinguishable by only the SNVs but not CNAs. Thus in experiment 13, we select a percentage of internal nodes, a variable that we vary in this experiment, so that these internal nodes do not have any CNAs on their outgoing edges.

Based on the true tree structure and the CNAs on the true tree, we then form the **true CNA tree** as follows. First, we identify the edges that do not have any CNAs on the true tree, and remove these edges along with the child nodes these edges are connecting with. The removed nodes' child nodes, if any, are connected with the removed nodes' parent nodes. The resulting tree structure is the true CNA tree. Notice that in all experiments except experiment 13, every edge on the true tree has one CNA and thus is not subject to be removed. Thus in all experiments except experiment 13, the true CNA tree's structure is the same as that of the true tree.

Next, based on the true CNA tree structure, we simulate the **inferred CNA tree**, also called **CNA tree** in the main text. Here we use the terminology "inferred CNA tree" to distinguish it from the "true CNA tree". The inferred CNA tree is one of the inputs to SCsnvcna. We simulate the inferred CNA tree by simulating both the inferred CNA tree structure and the CNA cell CPs on the leaf of the inferred CNA tree. The inferred CNA tree structure differs from that of the true CNA tree by the unobserved CNAs and the missing leaf nodes.

For all experiments except experiment 14 (**Supplemental Table S1**, **Supplemental Methods Section 2.5**), we assume that there is no unobserved CNA subclones, and thus the inferred CNA tree structure is the same as the true CNA tree structure. However, in the real case scenario, it is possible that a certain percentage of CNA leaf nodes are unobserved or missing due to the sampling bias of the CNA cells. Thus in experiment 14, we sample the leaf nodes from the true CNA tree and remove them based on their CPs. The higher the CP, the more likely the leaf node is selected to be removed to mimic the real case scenario when the leaf nodes on the CNA tree might be missing due to sampling bias. We varied the percentage of leaf nodes that are missed to investigate SCsnvcna's performance when different degrees of leaf nodes are missing in this experiment. Next, we simulate the CNA cell CPs on the inferred CNA tree structure, focusing on the leaf level. For all experiments except experiment 12 (**Supplemental Table S1**, **Supplemental Methods Section 2.3**), we simulate CNA cells' CPs with the same value as the true CPs of the corresponding nodes on the true tree. Thus in all experiments except experiment 12, the difference between SNV cells' CPs and CNA cells' CPs is the same difference between SNV cells' CPs and true CPs. However, in the real case scenario, it is possible that the CNA cells have a sampling bias from the true tree. Therefore, in experiment 12, we sample CNA cells' CPs on the leaves of the inferred CNA tree by a multinomial distribution from the corresponding true CPs. Notice that since SCsnvcna requires only the inferred CNA tree structure and the CPs for each leaf node on the inferred CNA tree as the input, we did not simulate the actual CNA cells or the number of CNA cells.

The resulting tree is the inferred CNA tree. The inferred CNA tree structure along with the CNA cells' CPs is one of the inputs to SCsnvcna.

Second, starting from the true tree, we simulate the SNVs on the branches and SNV cells on the leaf nodes while considering mutation loss and CP difference between SNV cells and CNA cells. In particular, given a pre-determined SNV number that can be varied (Experiment 6 in **Supplemental Table S1**), we evenly distribute the SNVs on the branches of the true tree according to the branch lengths. To simulate the placement of SNV cells, we distribute a pre-determined SNV cell number that can be varied (Experiment 5 in **Supplemental Table S1**) on the leaf nodes on the true tree according to the interval length that each leaf node has while considering σ', the standard deviation of the CPs between SNV cells and CNA cells (Experiment 10 in **Supplemental Table S1**). In particular, the percentage of SNV cells to be placed on a certain leaf node follows a Gaussian distribution whose mean equals the interval length that the leaf node has, and standard deviation is σ'. In our experiment 11 in **Supplemental Table S1** (mean of a Beta distribution imputed to CP difference), in addition to a Gaussian distribution, we further increased/decreased SNV cells' CP by a percentage from a Beta distribution, as described in **Supplemental Section 2.2**, so as to further enlarge the CP difference between the SNV cells and the CNA cells. Since it is possible that the CNA cell CPs may differ from the true CP by a non-Gaussian distribution, in experiment 12, we sampled both SNV cells' and CNA cells' CPs from the true CP according to a multinomial distribution. Once the SNVs are placed on the edges and SNV cells are placed on the leaf nodes, we simulate the underlying genotype matrix G such that $G[i, j] = 1$ if mutation j appears on the path between root and the leaf node where cell i is located. Otherwise $G[i, j] = 0$. We then impute mutation losses and modify G matrix accordingly. To achieve this, we randomly select pairs of SNVs and CNAs whereas the edges the CNAs are located are limited to those on the path between the SNV and one of its descendants. This is to make sure that the SNV can be lost due to a CNA loss. The number of pairs of SNVs and CNAs is determined according to the mutation loss rate (Experiment 8 in **Supplemental Table S1**) such that the total number of pairs equals the total number of edges multiplied by the mutation loss rate. If a SNV appears in more than one pair, we randomly select one and eliminate the rest. This is due to that although it is possible for a SNV to be lost due to parallel copy number losses, the possibility is low in biology. Once a SNV is determined to be lost on a certain paired CNA, we flip from 1 to 0 all the entries of this SNV on the G matrix for the cells which are the descendants of the edge where the CNA is located. Note that before this step, whether a CNA is a gain or loss was not determined because we did not simulate the actual CNA but focused on the CNA tree structure. This is why the pair-up between SNVs and CNAs is random, not limiting CNAs to a certain subset. Our goal is to examine that based on the D matrix that contains such mutation loss signal, whose simulation process is described below, whether SCsnvcna is able to correctly place SNVs and SNV cells, as well as detect mutation losses.

Based on the G matrix, we create the D matrix by imputing false positive entries, false negative entries and missing entries. In particular, we randomly flip the entries in G matrix to be the missing value 3 according to the variable missing rate (Experiment 3 in **Supplemental Table S1**). We then randomly flip the entries whose values are 0 to be 1 according to the variable false positive rate (Experiment 1 in **Supplemental Table S1**), and vice versa for the variable false negative rate (Experiment 2 in **Supplemental Table S1**).

In the real case, even though SNV cells are not designed to detect CNAs, it is possible that some large CNAs are detectable by SNV cells. We want to evaluate through simulation how much such a CNA signal from SNV cells can help SCsnvcna to correctly place the SNVs on the CNA tree, given different degrees of CNA signals that a SNV cell can detect. To simulate this, we select a certain percentage of the CNAs and determine that they are detectable by the SNV cells that have them. Given this information, SCsnvcna will constrain the placement of the SNV cells that have these CNAs to those CNA leaf nodes that also contain these CNAs. The percentage of the CNAs selected was a variable that we varied in experiment 7 shown in **Supplemental Table S1**.

The last step of the simulator is to simulate a list of potential overlapping SNVs and copy number losses. While in the previous steps we imputed mutation losses, this step is to mimic the real case scenario when we do not have the list of actual mutation losses but the list of pairs of overlapping SNVs and copy number losses. Such a list provides SCsnvcna all the SNVs that have the potential to be lost at certain edges whereas leaving the decision of whether each SNV is truly lost or not to SCsnvcna. For real data, such a list is obtained by simply comparing the genomic locations of all SNVs and all copy number losses in a pairwise manner. Since our simulator does not involve genomic locations, to simulate such a list, we randomly select half of the edges to pair up with a SNV that has never been lost and add such pairs to the list. We also add to the list the pairs of the SNVs and copy number losses in which SNVs have truly been lost. Thus such a list contains both the bona fide SNV losses and the SNVs that are not lost despite their overlapping with some copy number losses.

2 Design of simulation experiments

2.1 Overview

Fifteen simulated datasets, each with a varying variable, were generated to fully test SCsnvcna under different conditions and compare it with other seven methods. For each simulated dataset, we fixed all other variables to their default values while varying one variable. **Supplemental Table S1** summarized these 15 datasets by specifying their group, the varying variables for each dataset (second column) and the varying values (third column). The number with "d" on the right in the parenthesis is the default number for this variable. In the following, we describe in detail how we simulated experiments 11-15.

2.2 Experiment 11: Mean of a Beta distribution imputed to CP difference

Since in reality, the CP difference between SNV cells and CNA cells may not be according to a Gaussian distribution, to test how SCsnvcna would behave when the CP difference of the subclones fluctuates more than what a zero-mean Gaussian may render, we designed the experiment such that the CP difference was not according to a zero-mean Gaussian distribution. In particular, we randomly selected a node on the ground truth tree structure, and increased or decreased this node's SNV cells' CP randomly by a ratio. The ratio was determined by a sampling from a Beta distribution, and the mean of the Beta distribution varied from 0 to 0.8. Notice that before this step, the selected node's CP of SNV cells by default already differed from its corresponding CP of CNA cells by a Gaussian distribution. Thus this node's CP of SNV cells and CNA cells differed not only by a Gaussian distribution with zero mean, but also a ratio from a Beta distribution. Note that the selected node could be either an internal node or a leaf node. To make sure that the perturbation of the CP was large enough to mimic the large sampling bias between SNV cells and CNA cells, we selected only those nodes whose SNV cells' CP was ≥ 0.2. Such a perturbation was reflected in all descendant leaf nodes of this node, i.e., all descendant leaf nodes' CP would increase or decrease by the same ratio. The perturbation would also happen to the leaf nodes that were not the descendant of the selected node so that the total SNV cells' CPs of all leaves can still add up to 1. In more detail, if the selected node's CP increased by x, then the CPs of the leaf nodes that were not the descendant of the selected nodes would decrease by $(1-(1+x)*c)/(1-c)$, in which c was the original CP of the selected node.

2.3 Experiment 12: Both SNV/CNA cells' CPs sampled from a multinomial distribution from true CPs

This experiment is to mimic the situation when both SNV cells' CPs and CNA cells' CPs differ from that of the true tree, and the difference is not according to a Gaussian distribution. We aimed at testing SCsnvcna's robustness in the face of such a situation in this experiment. We generated the simulated data by sampling both CNA cells' CPs and SNV cells' CPs from the true CP by a multinomial distribution for each subclone on the leaf level. Notice that these two sets of cells' CPs were sampled from that of the true tree independently, which made their CPs differ from each other more than the case when SNV cells' CPs were drawn directly from the CNA cells' CPs.

2.4 Experiment 13: Percentage of internal nodes which do not have CNAs on the outgoing edges on the true tree

To quantify how SCsnvcna performs when some edges on the true tree do not have CNAs in terms of the genotype error, pairwise SNV error and pairwise SNV and CNA error, we randomly selected a certain percentage of internal nodes (0%, 10%, 25% and 50%) on a true tree and determined that there was no CNA on their outgoing edges. We then form the true CNA tree from such a true tree by connecting the grandchild nodes of the selected internal nodes with these selected internal nodes, as also described in Supplemental Methods Section 1. Therefore, all those edges where there were no CNAs were removed in the true CNA tree. Thus for a complete binary tree with 16 leaves (thus 15 internal nodes), if 50% of the internal nodes were selected, in total half of the nodes (including both internal and leaf nodes) would be removed in the true CNA tree. In this experiment, we assumed all the CNAs can be detected so that the inferred CNA tree (the input of SCsnvcna) has the same structure as the true CNA tree. The resulting inferred CNA tree structure is the input CNA tree to SCsnvcna.

2.5 Experiment 14: Percentage of missing leaf nodes on the CNA tree

This experiment focuses on the performance of SCsnvcna in the face of imperfect inferred CNA tree due to sampling bias. Notice that in experiment 13, the true CNA tree's structure differs from the true tree due to the lack of CNAs to distinguish two subclones. In this experiment, the true CNA tree's structure is the same as the true tree's structure, but the inferred CNA tree's structure differs from the true CNA tree's structure due to missing CNA leaf nodes resulting from sampling bias of CNA cells. Thus in experiment 13, we focused on the case when there lacked CNAs biologically distinguishing two subclones, whereas in this experiment, we focused on the case when some CNA subclones did occur but were not observed during the sequencing process.

In more detail, sampling bias of the CNA cells may occur during sequencing and lead to missing leaf nodes on the CNA tree and thus change the CNA tree's structure. Subclones with a lower CP are more subject to be missed due to that they are represented by a smaller number of cells. To test SCsnvcna's performance in the presence of missing leaf nodes in the CNA tree, we removed a certain percentage of leaves on the CNA tree. Such a percentage was a variable that we varied in this experiment (0%, 10%, 25% and 50%). Each leaf node's chance to be selected was calculated as a normalized inverse of the CP that the node has. In this way, the lower the CP, the higher the chance the node may be selected, yet all leaf nodes' chances being selected added up to 1. The resulting tree structure, the inferred CNA tree with missing leaf nodes, was the input CNA tree to SCsnvcna. In evaluating the error rates, the placement of the SNVs was mapped back to the corresponding edges on the true tree, and all CNAs including those that were missing due to missing leaf nodes were involved in evaluating the pairwise SNV/CNA errors.

2.6 Experiment 15: Number of bulk samples given a CNA bulk tree

Since SCsnvcna takes a CNA tree as the input, we tested how SCsnvcna would perform given a bulk tree instead of a single-cell CNA tree. In more detail, given the ground truth single-cell CNA tree that had 16 leaves, we sampled X bulk samples, in which $X = 4$ and 8, from the leaf nodes of the CNA tree with replacement. To mimic the process of bulk sampling, bulk samples were sampled based on the percentage of cells each leaf node of the given tree had. In reality, such a sampled bulk tissue may not be representing only one subclone but a combination of multiple subclones, depending on the size of the resected tissue and the distribution of the subcloens. Here we simplified the process by assuming that a bulk sample represented only one subclone because the deconvolution of the bulk sample that contains multiple subclones is another big topic and is out of the scope of this study. We then built a bulk tree such that the leaf nodes represented the sampled bulk tissues and their CPs were proportional to the number of times they were sampled. We removed all the edges and nodes that were not on a path to the resulting leaf nodes. We further merged internal nodes if they were all on a path that did not branch. We then ran SCsnvcna given the resulting bulk tree.

3 Comparison with SCARLET

3.1 Differences between Scarlet and SCsnvcna

SCARLET(Satas et al., 2020) infers the placement of SNVs on a tree by considering the potential loss of the SNV when it overlaps with a copy number loss. It assumes that both the SNVs and CNAs are known on the same cell, and that all copy number profiles, whether they are on the internal nodes or leaves, are represented by cells sequenced.

SCsnvcna differs from SCARLET in the following ways.

- SCsnvcna assumes that the SNV cells and CNA cells are two different groups of cells sampled from the same tumor, whereas SCARLET assumes the same set of cells provides both SNVs and CNA calls.
- SCsnvcna does not require the internal nodes on the CNA tree to be represented by SNV or CNA cells, whereas by design SCARLET assumes all nodes on the CNA tree, whether internal nodes or leaves, are represented by cells sequenced.

The first difference is a major difference between SCsnvcna and SCARLET, as without the cells constraining the path of the SNVs, the search space of SNV placement, true genotype matrix, as well as SNV cell placement is huge. Thus SCsnvcna is solving a completely different problem from SCARLET. The huge search space also raises up the challenge that SCsnvcna faces. To make a fair comparison with SCARLET, in our simulation experiments 1-6 and 8-9, we fixed all SNV cells' placement to the correct leaf nodes and thus do not search for the placement of SNV cells. Instead, we focused on the search of the SNV placement, the underlying genotype matrix and the latent variables. However, for the rest of the experiments, i.e., experiments 7 and 10-15, we also showed the results when the SNV cells' placement was not fixed to demonstrate that SCsnvcna is capable of correctly placing the SNV when SNV cell placement is unknown.

For the second difference, SCsnvnca does not assume that the internal nodes on the CNA tree may be represented by any cells to be sequenced. This is more realistic than SCARLET because it is possible that the ancestral nodes are not represented by any cells at the time when the tumor cells are sampled and sequenced.

3.2 Simulating SCARLET's input data

Since SCARLET's input data is different from those of SCsnvcna's, we simulated the input files for SCARLET separately but from the same CNA tree, SNV placement and SNV cell placements. Specifically, SCARLET requires that each node, both internal node and leaf node, has at least one cell attached to it. We therefore added one cell to each node in the CNA tree if there had not been any cell assigned to the node yet. Notice that SCARLET assumes that the SNVs and CNAs come from the same set of cells and thus does not distinguish SNV cells and CNA cells. We then created the expanded D and G matrices that had the extra rows corresponding to the extra cells attached to the nodes on the tree. The extra rows in the expanded D matrix had the same missing rate, false positive rate and false negative rate as defined by the simulation experiment. We also provided SCARLET the potential mutation loss list that contained the pairs of SNVs and edges where CNAs were placed due to which a mutation loss may occur.

Since SCARLET takes read counts instead of the cell by SNV matrix D as the input, we converted the expanded D matrix to total read count (T) and variant read count (V) by Eq. 1.

$$T, V = \begin{cases} Poisson(d), Poisson(d/2), & \text{if } D_{i,j} = 1 \\ Poisson(d), Poisson(\epsilon), & \text{if } D_{i,j} = 0 \\ Poisson(\epsilon), 0, & \text{if } D_{i,j} = 3 \end{cases} \tag{1}$$

where d represents read depth at the SNV site. In our simulation, we set $d = 100$ and $\epsilon = 0.001$. Such a Poisson sampling of the read counts can also be found in SCARLET's simulation experiments(Satas et al., 2020). We acknowledge that to generate the simulation data for SCARLET, there is an additional sampling error (Poisson sampling of the sequencing reads) involved, and such an additional error might be one of the causes of the inferior performance of SCARLET to SCsnvcna.

4 Simulating input data for SiFit, SiCloneFit, SCG, RobustClone, Compass, and BitSC^2

4.1 Simulating SiFit's input data

SiFit(Zafar et al., 2017) is a statistical inference method that uses a finite-sites model to infer tumor phylogenies from noisy single-cell sequencing data. We transposed SCsnvcna's observed genotype matrix to prepare it for the analysis of SiFit. Additionally, SiFit also requires input of cell names and mutation names to infer the mutation order. To facilitate this, we generated names for each simulated cell and SNV.

4.2 Simulating SiCloneFit's input data

SiCloneFit (Zafar et al., 2019) is a tool based on Bayesian principles that enables simultaneous estimation of tumor clones, clonal genotypes, and clonal phylogeny using the mutation profiles of single cells, even in the presence of measurement noise. Since SiCloneFit takes the same input as SiFit, the simulation input data for SiFit was used directly for SiCloneFit.

4.3 Simulating SCG's input data

SCG(Roth et al., 2016) is a sophisticated single-cell genotyping tool capable of accurately inferring clonal genotypes. This state-of-the-art software requires as input a genotype matrix, which is then used to cluster cells and infer clonal genotypes. With some minor modifications to the header information, SCG can seamlessly take our simulated genotype matrix developed for SCsnvcna.

4.4 Simulating RobustClone's input data

RobustClone (Chen et al., 2020) was reported to be able to accurately retrieve the genuine genotypes of subclones by performing a low-rank matrix decomposition. RobustClone accepts a genotype matrix as input. In the case of our simulated D matrix, it can be readily utilized with minor formatting adjustments for RobustClone.

4.5 Simulating Compass' input data

To generate Compass' input data, we assumed each edge on the simulation tree represented a CNA, and we ensured that there was no overlap between regions assigned to each CNA. Subsequently, single nucleotide variations (SNVs) were randomly assigned to a CNA region and a reference base was assigned. To simulate copy number aberrations for each region, we randomly sampled the copy number from a range of 1 to 5. In the event that SNVs were lost on the tree, we set the corresponding regions to have a copy number of 1 for the cells that had experienced the loss. We then used Poisson distribution to simulate the reference count and variant read count. we converted the D matrix to reference read count (R) and variant read count (V) by Eq. 2.

$$R, V = \begin{cases} Poisson(d * (CN - 1)), Poisson(d), & \text{if } D_{i,j} = 1 \\ Poisson(d * CN), Poisson(\epsilon), & \text{if } D_{i,j} = 0 \\ Poisson(\epsilon), 0, & \text{if } D_{i,j} = 3 \end{cases} \tag{2}$$

where d represents read depth at the SNV site for one copy and CN is the copy number for the region where SNV is placed on. In our simulation, we set $d = 20$ and $\epsilon = 0.001$. Since COMPASS does not only require the read count for each SNV, but also the total read count for each region, we simulated such a read count, T, using Poisson distribution which considers the copy number CN at the region, the read depth d for one copy, and the region size (Eq. 3). Here we use $S + 1$ to represent region size in which S is the number of mutations so that when there is zero mutations, the total read count remains nonzero.

$$T = Poisson(d * CN * (S + 1)) \tag{3}$$

4.6 Simulating BitSC^2's input data

BitSC^2 takes the total read counts and variant read counts for each mutation site. We thus used Poisson distribution to simulate the total count and variant read count. We converted the D matrix to total read count (T) and variant read count (V) by Eq. 4.

$$T, V = \begin{cases} Poisson(d * CN), Poisson(d * CN/2), & \text{if } D_{i,j} = 1 \\ Poisson(d * CN), Poisson(\epsilon), & \text{if } D_{i,j} = 0 \\ Poisson(\epsilon), 0, & \text{if } D_{i,j} = 3 \end{cases} \tag{4}$$

where d represents read depth at the SNV site for one copy and CN is the copy number for the region where SNV is placed on. In our simulation, we set $d = 20$ and $\epsilon = 0.001$.

5 Simulated datasets using a different simulator from the simulator proposed in this study

This experiment is to test SCsnvcna's robustness when an orthogonal simulator is used to simulate the data. We chose the simulator published along with COMPASS Sollier et al. (2023) because it simulates both SNVs and CNAs on single cells, as well as considers the allele dropout and loss of heterozygosity, the variables that make the simulated data more like the real datasets. We fixed the CNA number to be 30, region number to be 100, and varied three variables one at a time and thus generated three datasets. For the first dataset, we varied the number of cells from 100 (default), to 200, to 500. For the second dataset, we varied the number of mutations from 50, to 100 (default), to 200. For the third dataset, we varied the number of nodes from 8 (default) to 16. Since only one variable was varied at a time, variables were set as default when they were not varied. We repeated each variable setting for five times to avoid extreme cases. In running SCsnvcna, we used the simulated CNA tree as one of the inputs to SCsnvcna. Also notice that we input BiTSC^2 with the correct number of subclones like what we did in our previous experiments.

6 How to correct ploidies without knowing the correct copy number profiles?

Due to the limitation of the existing CNA detection tools, ploidies of individual cells may not be estimated correctly. This further leads to the wrong copy number profile for the entire genome. To correct the ploidies without the prior knowledge of the copy number profile, users can cluster the cells according to breakpoints using a combination of UMAP McInnes et al. (2018) for dimension reduction and HDBSCAN McInnes et al. (2017) for clustering, the latter of which does not require the prior knowledge of the number of clusters. The clusters of the cells indicate the subclonality of the cells, i.e., cells clustered together shall belong to the same subclone, and thus have very close ploidies. In this way, users can identify cells with outlier ploidies for each subclone. Users can further identify whether a certain cell's ploidy has been elevated (detected ploidy is much larger than other cells' ploidies in the same cluster) or decreased (detected ploidy is much smaller than other cells' ploidies in the same cluster) by the CNA detection tool.

Users can either remove these cells from downstream analysis in building the CNA tree, or correct these cells' absolute copy number. To do the latter, use the following equation to find the inferred absolute copy number that should have been inferred as 2.

$$\frac{2}{coP} = \frac{Y}{inP},$$

in which coP is the correct ploidy (the ploidy from the majority of the cells in the same cluster such as median of the ploidies of the cells in the same cluster), and inP is the incorrectly inferred ploidy for a particular cell in the same cluster. We then find the closest integer to Y, which is the absolute copy number inferred by the CNA caller that should have been inferred as 2. For example, suppose $coP = 3$, and $inP = 5$, then $Y = 2/3 * 5 = 3.33$. The closest integer to 3.33 is 3. Thus for a cell whose ploidy was inferred wrongly as 5, for the regions whose absolute copy number was inferred as 3, it should be corrected to be 2. Any region with absolute copy number > 3 is a copy number gain, and any region with absolute copy number < 3 is a copy number loss. To further obtain the absolute copy number for the entire genome instead of just acquiring the copy number gain/loss status, the following formula can be used:

$$X/coP = Y/inP,$$

in which $X = 1, 3, 4, 5, \ldots$.

Just as how we found the corresponding copy number to 2 in the cell with incorrectly inferred ploidy, we can correct all the nearest integer values of Y (wrong copy number) to their corresponding X (correct copy number). By doing this, the whole absolute copy number profile as well as ploidy can be corrected.

7 Tuning the threshold of the percentage of CNAs specific to a CNA leaf that a SNV cell contains for the SNV cell to be constrained

In the subsection **Using the CNAs on SNV cells to limit the placement of SNV cells** in **Methods**, the second strategy described how we utilize CNAs specific to a CNA leaf to constrain a SNV cell. Specifically, for a SNV cell to be constrained to a CNA leaf, the SNV cell shall contain $> x\%$ of the CNAs that are specific to the CNA leaf. The selection of the threshold x affects the number of SNV cells that can be constrained as well as the number of CNA leaves a SNV cell can be constrained to. To investigate how the latter two numbers vary with the selection of x, we tuned x between 50 and 90, and reported 1) the percentage of the SNV cells being constrained; and 2) average percentage of CNA leaves a SNV cell is constrained to in **Supplemental Table S2**. As expected, the higher the x, the smaller percentage of SNV cells that can be constrained. We also found that higher x leads to lower percentage of CNA leaves a SNV cell is constrained to, and thus results in a more specific placement of a SNV cell. In summary, there is a tradeoff between the two factors: the specificity of the CNA leaves that a SNV cell is constrained to and the percentage of SNV cells that can be constrained at all. In practice, we recommend a relatively big value of x because of the benefit of the high specificity of the CNA leaves that the SNV cells shall be constrained to, such as $> 80\%$. For those SNV cells not meeting the criterion to be constrained, SCsnvcna will use the MCMC sampling to search for the optimal placement based on the mutation genotype matrix and SNV cells' CPs.

References

Chen Z, Gong F, Wan L, and Ma L. 2020. Robustclone: a robust pca method for tumor clone and evolution inference from single-cell sequencing data. *Bioinformatics* **36**: 3299–3306.

Mallory XF and Nakhleh L. 2022. Simscsntree: a simulator of single-cell dna sequencing data. *Bioinformatics* **38**: 2912–2914.

McInnes L, Healy J, and Astels S. 2017. hdbscan: Hierarchical density based clustering. *J. Open Source Softw.* **2**: 205.

McInnes L, Healy J, and Melville J. 2018. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426* .

Roth A, McPherson A, Laks E, Biele J, Yap D, Wan A, Smith MA, Nielsen CB, McAlpine JN, Aparicio S, et al.. 2016. Clonal genotype and population structure inference from single-cell tumor sequencing. *Nature methods* **13**: 573–576.

Satas G, Zaccaria S, Mon G, and Raphael BJ. 2020. Scarlet: Single-cell tumor phylogeny inference with copy-number constrained mutation losses. *Cell Systems* **10**: 323–332.

Sollier E, Kuipers J, Takahashi K, Beerenwinkel N, and Jahn K. 2023. Compass: joint copy number and mutation phylogeny reconstruction from amplicon single-cell sequencing data. *Nature Communications* **14**: 4921.

Zafar H, Navin N, Chen K, and Nakhleh L. 2019. Siclonefit: Bayesian inference of population structure, genotype, and phylogeny of tumor clones from single-cell genome sequencing data. *Genome research* **29**: 1847–1859.

Zafar H, Tzen A, Navin N, Chen K, and Nakhleh L. 2017. Sifit: inferring tumor trees from single-cell sequencing data under finite-sites models. *Genome biology* **18**: 1–20.